1. Einführung in AMD Vivado

Diese Anleitung zeigt, wie Sie die AMD Vivado Umgebung für die FPGA Entwicklung einsetzen können.

Begriffsbestimmungen und Abkürzungen:

FPGA	Field Programmable Gate Array, ein programmierbarer Baustein.
Xilinx	Xilinx Inc. ist der weltgrösste Entwickler und Hersteller von FPGA
Vivado	Die aktuelle Software von Xilinx für CPLD und FPGA Entwicklungen
VHDL	Very High-Speed Hardware Description Language

2. Neues Projekt mit Vivado

2.1. Starten der Vivado Software

Es gibt verschiedene Möglichkeiten, Vivado zu starten:

- Suche nach *Vivado* in der Startleiste von Windows oder Linux
- Doppelklick auf ein bestehendes *.xpr* File (Xilinx Project File) eines Projektes

2.2. Erzeugung eines neuen Projektes

Nach dem Start von Vivado (ausser durch ein **.xpr* File) erscheint der Startbildschirm mit der Auswahl *Create Project*.

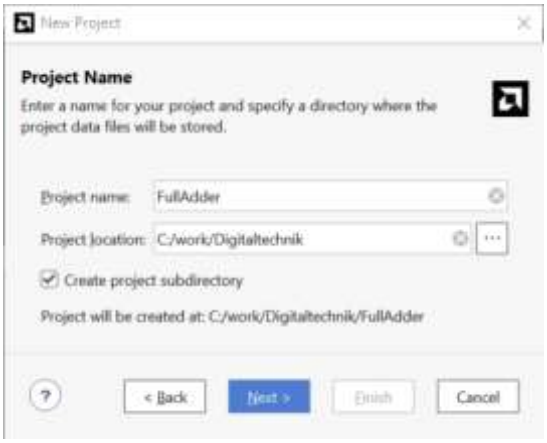
Man kann dann einfach auf diese Auswahl klicken, oder aber aus dem Menu heraus ein neues Projekt starten mit *File - New Project*….

Im neuen Fenster erscheint dann zuerst die Begrüssung durch den *Wizard*. Hier kann man einfach *Next* klicken.

2.2.1. Verzeichnis, Name und Top-Level Entity

Projektname: Wählen Sie ausdrucksstarke Namen, die klar sagen, was der Inhalt ist. Nicht einfach nur *project_1* …

Location: Es beschleunigt die Synthese enorm, wenn das Projekt auf einem lokalen Laufwerk ist.



Create project subdirectory: Unbedingt anwählen!

Achtung: Verzeichnispfad und Projektnamen dürfen keine Umlaute enthalten!

Dann *Next* klicken …

2.2.2. Projekt Typ

Im Rahmen der Unterrichtsaktivitäten werden wir immer *RTL Project* verwenden. Dann klicken Sie *Next*.
Ohne Haken bei *Do not specify sources at this time* werden Sie als nächstes nach einem oder mehreren Sourcefiles gefragt. Man könnte diese auch erst später zu einem Projekt hinzufügen.

2.2.3. Add Sources

Beim nächsten Fenster *Add Sources* können Sie Design-Dateien definieren.

Klicken Sie auf *Create File*, und geben Sie den neuen Namen mit Endung ein. Sie können hier beliebig viele Dateien hinzufügen.

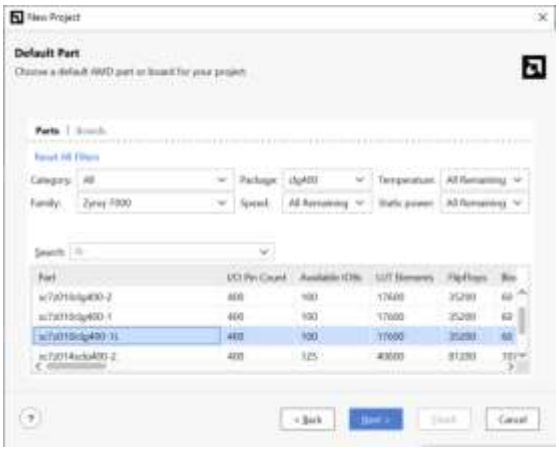
Kontrollieren Sie, dass am unteren Rand die *Target Language* und auch *Simulator Language* beide auf *VHDL* stehen. Dann klicken Sie *Next*.

2.2.4. Add Constraints

Hier geht es um die «Constraints» beziehungsweise Randbedingungen für das Projekt. Dies ist dann speziell wichtig, wenn das Design über Pins des FPGA mit der Aussenwelt verbunden sein soll.
Zurzeit benötigen wir das jedoch nicht, und können diesen Schritt überspringen.

2.2.5. Auswahl des Bauteils

Auch wenn wir unser Design noch nicht auf ein FPGA bringen wollen, müssen wir hier trotzdem bereits den richtigen Chip auswählen. Am einfachsten wählen Sie bei *Family* die *Zynq-7000* Serie an, und bei Package das Format *clg400* (0.8mm Pitch, RoHS (= bleifrei) kompatibles Gehäuse mit 400 Pins).
Mit dieser Einstellung erscheint dann in der *Part* Auswahl unser Chip *Xc7z010iclg400-1L* zuoberst in der Liste und kann dann leicht angewählt werden.
«xc» steht dabei für Xilinx Commercial,
«7z» ist die Virtex 7000 Zynq Generation
«010» ist eine Angabe für die Grösse (Zahl Logik-Elemente im FPGA)
«i» steht für die Temperaturfestigkeit (Industrial, -40° bis +100°C)
«clg» steht für das Gehäuse, mit 0.8mm Pitch und intern «wire-bond» und somit noch nicht «flip-chip».
«400» ist die Zahl der Pins, also ein Array von 20 x 20 Pins
«-1L» ist der «Speed Grade». «1L» ist langsam, aber Low-Power fähig.



Diese Detailinformationen sind zurzeit nicht wichtig und sie können nach der Auswahl des richtigen Bauteils einfach *Next* drücken.

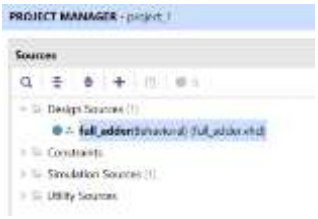
2.2.6. New Project Summary

Hier werden die gewählten Einstellungen nochmals zusammengefasst. Klicken Sie einfach *Finish*.

Nach dem Finish kommt ein Fenster zur Definition der Module. Hier könnte man interaktiv die Ports für das neue Design-File definieren. Das ist unpraktisch. Überspringen Sie diesen Schritt einfach mit *OK* und dann *Yes*.

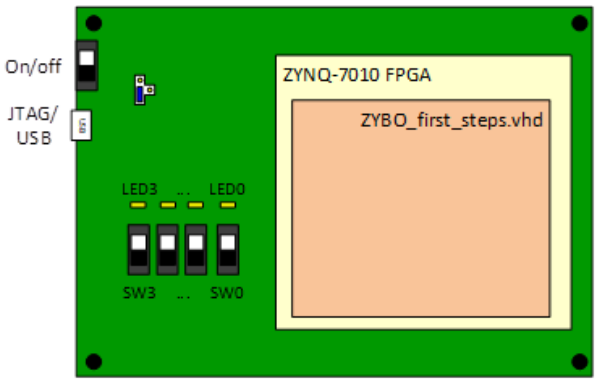
2.2.7. Editieren der Files

Jetzt kann es losgehen. Im *PROJECT MANAGER* Fenster sieht man die Files und Einheiten (ENTITY) des Designs. Design-Elemente erscheinen gleichzeitig immer bei *Design Sources* und auch bei *Simulation Sources*. Durch Doppelklick öffnet man die Files zum Bearbeiten.



3. Erstes Vivado Projekt in VHDL mit ZYBO Board

In diesem Kapitel geht es darum, eine möglichst einfache Schaltung auf den Zynq-FPGA des ZYBO Board von Digilent zu bringen, um den ganzen Design-Fluss einmal durchzuspielen.
Die VHDL Schaltung ist sehr simple und steuert mit den 4 Schaltern synchron zum 125 MHz Takt die 4 LEDs an.



3.1. Neues Projekt «ZYBO_First_Steps»

Erzeugen Sie, wie bereits im vorhergehenden Kapitel erklärt, ein neues Projekt, diesmal jedoch mit dem Namen *ZYBO_First_Steps*. Gehen Sie so vor wie im letzten Kapitel beschrieben.

Bei *Add Sources* erzeugen Sie ein neues File mit dem Namen *ZYBO_First_Steps.vhd*. Und bei *Add Constraints* erstellen Sie ein neues File Xilinx Design-Constraints (xdc) File mit dem Namen «ZYBO_First_Steps.xdc». Dieses wird die Pin-Belegung und weitere Elemente zur Steuerung der Synthese enthalten.
Das FPGA Bauteils wählen Sie mit *xc7z010clg400-L1*.

3.2. VHDL Design File

Öffnen Sie im Project Manager unter Design Sources (1) das VHDL File ZYBO_First_Steps.vhd mit Doppelklick. Ersetzen Sie darin alle von Vivado vorgegebenen Zeilen mit dem Inhalt wie unten gezeigt. Sie können die paar Zeilen abtippen oder mit Copy-Paste in Vivado übernehmen.
Was macht der Code? Bei jeder positiven Flanke von *isl_clock* werden die 4 Schaltereingänge *islv4_switch* auf die 4 LED Ausgänge *oslv4_led* kopiert.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY zybo_first_steps IS
    PORT (
        isl_clock     : IN  STD_LOGIC;
        islv4_switch  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
        oslv4_led     : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
    );
END ENTITY zybo_first_steps;


ARCHITECTURE rtl OF zybo_first_steps IS

BEGIN

    reg_proc : PROCESS (isl_clock)
    BEGIN
        IF rising_edge(isl_clock) THEN
           oslv4_led   <= islv4_switch;
        END IF;
    END PROCESS reg_proc;

END ARCHITECTURE rtl;
```

Speichern Sie Ihre Änderungen in Vivado mit *Save File*.

3.3. Pin-Definition im .XDC File

Damit Vivado die Ein- und Ausgänge richtig verbinden kann, müssen diese irgendwo definiert werden. Dies geschieht im XDC-File (Xilinx Design-Constraints) – das sind die Constraints oder Randbedingungen. Öffnen Sie im *Project Manager* unter *Constraints (1)* das XDC File *ZYBO_first_steps.xdc* mit Doppelklick. Ersetzen Sie allfällig vorgegebenen Text mit den folgenden Zeilen:

```
##  Clock signal
##==============
set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { isl_clock }];
create_clock -add -name sys_clk_125m -period 8.00 -waveform {0 4} [get_ports {isl_clock}];
set_input_jitter [get_clocks -of_objects [get_ports isl_clock]] 0.1

##  Switches
##==========
set_property -dict { PACKAGE_PIN G15 IOSTANDARD LVCMOS33 } [get_ports { islv4_switch[0] }];
set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { islv4_switch[1] }];
set_property -dict { PACKAGE_PIN W13 IOSTANDARD LVCMOS33 } [get_ports { islv4_switch[2] }];
set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { islv4_switch[3] }];

##  LEDs
##======
set_property -dict { PACKAGE_PIN M14 IOSTANDARD LVCMOS33 } [get_ports { oslv4_led[0] }];
set_property -dict { PACKAGE_PIN M15 IOSTANDARD LVCMOS33 } [get_ports { oslv4_led[1] }];
set_property -dict { PACKAGE_PIN G14 IOSTANDARD LVCMOS33 } [get_ports { oslv4_led[2] }];
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33 } [get_ports { oslv4_led[3] }];
```

Sie erkennen jeweils den Zusammenhang zwischen dem *PACKAGE_PIN*, dem Standard für die I/O Spannung (Low-Voltage CMOS 3.3V) und den Signalnamen. Speichern Sie Ihre Änderungen in Vivado mit *Save File*.

3.4. Kompilierung

Die Umwandlung unserer Design-Eingaben in das FPGA Konfigurationsffile verläuft über mehrere Stufen. Diese können nacheinander einzeln gestartet werden, oder als Ganzes durch Doppelklick des dritten Knopfes *Generate Bitstream*.

3.4.1. Run Synthesis

Hier wird das Design analysiert und in ein internes Meta-Format umgewandelt, welches die gewünschten Strukturen und Zusammenhänge effizient abspeichert. Wenn es im Design Fehler gibt, dann werden Sie hier offensichtlich.
Dieser Schritt (wie auch die folgenden) dauert seine Zeit. Dabei wird rechts oben mit einem grünen rotierenden Ring angezeigt, dass der Prozess im Hintergrund läuft.

3.4.2. Run Implementation

Dabei wird das analysierte Design auf die im FPGA vorhandenen Strukturen und Gatter abgebildet. Dies beinhaltet die logischen Gatter, aber auch Multiplizierer und Speicher. Wenn es im gewählten FPGA zu wenig Ressourcen gibt, dann wird dies hier erkannt.

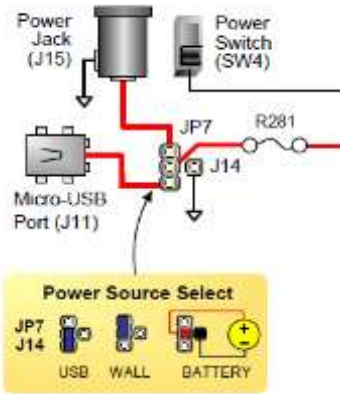
3.4.3. Generate Bitstream

Jetzt muss noch das effektive Konfigurations-File erzeugt werden. In diesem Schritt werden die Schnittstellen nach aussen angeschlossen, und das Bitstream File geschrieben.
Nach jedem Teilschritt kommt ein Pop-Up Menu, wie es weitergehen soll: Ob das Design zur visuellen Kontrolle geöffnet werden soll, oder ob der jeweils nächste Schritt gestartet werden soll.

3.5. Transfer des Files auf das FPGA Board

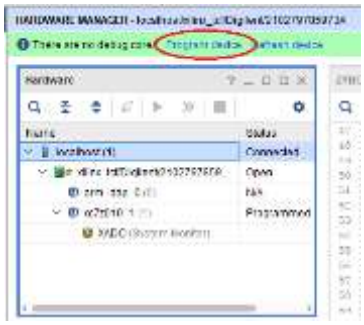
Bereiten Sie die FPGA Hardware vor:

- Kontrollieren Sie, dass der blaue 3-Positionen Jumper *JP7* auf *USB* steht.
- Schliessen Sie das Board über den Micro-USB Stecker mit der Bezeichnung *PROG UART* an den PC an.
- Schalten Sie das Board mit dem Schalter SW4 (beim USB-Stecker) ein.



Nach der Erzeugung des *Bitstream* kommt ein Pop-Up Fenster mit *Open Hardware Manager*. Wenn nicht, dann können Sie den Hardware Manager auch manuell öffnen, indem Sie im Menu unter *PROGRAM AND DEBUG* unter *Open Hardware Manager* diesen öffnen.
Mit *Open Target* und *Auto Connect* stellen Sie die Verbindung zur Ziel-Hardware her

Es erscheint neu das Fenster *Hardware Manager* mit dem genauen FPGA Chip und dem Link *Program device*.



Klicken Sie auf *Program device* und es erscheint ein weiteres Fenster mit dem automatisch gesetzten Bitstream-File und dem Knopf für *Program*.

Kontrollieren Sie, dass das Bitstream-File tatsächlich auf Ihr Projekt-Verzeichnis zeigt und starten Sie die Programmierung.

Wenn alles erfolgreich war, dann sollten Sie mit jedem der 4 Schaltern je eine LED ein- und ausschalten können.